REMARKS

Claims previously in issue:	1-27
Claims amended:	None
Claims cancelled:	None
Claims added:	None
Claims indicated as allowable:	4-7, 13-16, 22-25, if rewritten in independent form
Claims remaining in issue:	1-3, 8-12, 17-21, 26-27

The Invention

The invention includes a system, method, and program for parallelizing applications of certain script-driven software tools. In the illustrated embodiment, scripts in the software tool scripting language are automatically analyzed in order to produce a specification for a parallel computation which is functionally equivalent to the original script. The parallel computation specification is then executed by a parallel runtime system, which causes multiple instances of the original software tool and/or supplemental programs to be run as parallel processes. The resulting processes will read input data and produce output data, performing the same computation as was specified by the original script. The combination of the analyzer, runtime system, original software tool, and supplemental programs will, for a given script and input data, produce the same output data as the original software tool alone, but has the capability of using multiple processors in parallel for substantial improvements in overall "throughput".

The §103 Rejection

The Examiner has rejected claims -3, 8-12, 17-21, 26-27 under 35 U.S.C. §103(a) as being obvious over Andrews *et al.* 5,768,564 ("Andrews"). Applicant respectfully traverses this rejection with respect to the claims in issue.

Andrews is not relevant art. Andrews teaches "a method, system, apparatus, and program for translating one computer language to another using doubly-rooted tree data structures." Andrews, *Abstract*. Andrews teaches that source-to-source automatic translators are known, and use either of two distinct strategies to solve the problem of an unavailable compiler for a particular language on a particular computer. First, programmers may continue to write and maintain programs in the original source language. The translator converts these programs into intermediate code in a target language. An available compiler for the target language then converts this intermediate code into machine code which the target computer can understand. The translator does not have to produce readable or maintainable source code. The second strategy requires a translator to produce *readable and maintainable code*. Programmers going this route want to abandon the original language in favor of the target. The focus of Andrews's invention is this "more difficult task"; col. 1, l. 59 - col. 2, l. 8.

Andrews succinctly states its specific purpose:

> "A specific object of the present invention is to provide a method of translating code from a source computer language to a target *while preserving the preprocessor characteristics of the code as written in the source language.*" col. 2, ll. 60-63. [emphasis added]

These preprocessor characteristics of the source language code including macros, conditionally compiled regions of code, source inclusion statements, and comments. Andrews, *Abstract*.

Andrews neither teaches nor suggests anything about parallelizing a computer application program based on a script of a script-driven software tool. Indeed, the only mention found

of the word "parallel" in a computer search of the text of Andrews is a statement that "because of the opportunity for error inherent in manually maintaining parallel interfaces in both languages, it is essential that a product maintainer update just one copy of an interface and use a tool to generate the interface in the other language"; col. 11, ll. 43-48. Thus, Andrews uses "parallel" only in the context of explaining why maintaining two parallel sets of source code is error prone.

The Examiner points to Fig. 4, element 17 and to col. 6, ll. 36-41 in Andrews as teaching "a method for parallelizing a computer application program based on a script of a script-driven software tool". However, the figure and text simply indicate that Andrews's method behaves like "a normal compiler, with a few additional steps"; col. 6, ll. 33-34. Referring to Fig. 3, Andrews notes that:

> In the prior art compiler, source code 51 flows first through a scanner 53, where a stream of characters is divided into recognizable "words" or tokens and then to a parser 55, which determines a hierarchical relationship among the tokens and builds a syntax tree 57. An additional step, performed by a semantic analyzer 59, further refines the syntax tree 57. [Col. 19, ll. 15-22.]

The syntax tree 57 in Fig. 3 is single-rooted. Andrews's contribution to the art - the additional steps that Andrews references - is to use a doubly-rooted syntax tree, as shown in Figs. 2 and 4; col. 18, ll. 55-56.

Although the Examiner's arguments supporting the rejection are not wholly clear, it appears that the apparent depiction of parallelism shown in Fig. 4 by the source fragment tree 19, the

source syntax tree 7, and leaves 9 that join them is the basis for the Examiner's assertion that Andrews teaches parallelizing a computer application program. However, Fig. 9, an example of a fragment tree and syntax tree, clearly shows that the "parallel" leaves 9 are *simply parsed tokens derived from the first of two lines of expanded source code*; col. 21, ll. 27-47. Nothing is taught or suggested about parallelizing an entire computer application program.

Furthermore, while admitting that Andrews fails to teach "automatically analyzing the script and producing a parallel computation specification based on such analysis, where such parallel computation specification provides functional equivalence to the script when executed by a parallel runtime system", the Examiner asserts that these claim limitations are "implicitly" shown in Andrews. However, since Andrews does not even address the problem of parallelization of a computer application program of any kind, Andrews cannot fairly be said to teach or suggest such parallelization to one of ordinary skill in the art, or suggest to a skilled artisan how to adapt Andrews's techniques without undue experimentation to solve the problem addressed and solved by the present invention.

In short, Andrews teaches *nothing* about parallelizing a computer application program, let alone parallelizing a computer application program based on a script of a script-driven software tool. Accordingly, Andrews cannot serve as a basis of rejecting the claims as presented. The Examiner cites no additional support for his arguments (Andrews *et al.* 6,031,993 is simply a continuation of Andrews *et al.* 5,768,564).

Similarly, Andrews teaches *nothing* about producing a *parallel computation specification* based on analysis of a script of a script-driven software tool, where such parallel computation specification provides functional equivalence to the script when executed by a parallel runtime system.

Further, Andrews has no teaching or suggestion regarding automatically analyzing the script and producing a parallel computation specification *plus a script fragment set based on such analysis* (see claim 2 and similar claims). Simply put, Andrews does not deal with script fragments at all (see Application, p. 30, for a discussion on generation of script fragments in accordance with one embodiment of the invention).

Moreover, Andrews has no teaching or suggestion regarding *constructing a parallel dataflow graph from a serial dataflow graph* (see claim 3 and similar claims). Andrews simply does not deal with the concept of a parallel dataflow graph at all (see Application, Fig. 4 and text beginning at p. 8, l. 20, for a discussion on parallel dataflow graphs in accordance with one embodiment of the invention). The citation by the Examiner to Fig. 10 of Andrews is inapposite - that figure is not a dataflow graph as defined in the present invention, but simply the parsed tokens derived from second line of expanded source code of a particular example (col. 21, l. 27- col. 22, l. 23); the first line is shown in Fig. 9, discussed above. Fig. 11 is part of the same example.

The same arguments apply to the remaining rejected claims, many of which also distinguish from Andrews by the addition of additional limitations not taught or suggested by Andrews.

Conclusion

Accordingly, Applicant submits that none of the references, alone or in combination, anticipate or make obvious the invention as presently claimed. Applicant submits that this case is now in condition for allowance. Therefore, Applicant respectfully requests reconsideration and reexamination of the present application and allowance of the case at an early date.

Applicant asks that all claims be allowed. Enclosed is a $460.00 check for the Petition for Extension of Time fee. Please apply any other charges or credits to Deposit Account No. 06-1050.

Respectfully submitted,

Date: 3 JAN 2002

Kenyon S. Jenckes
Reg. No. 41,873

Fish & Richardson P.C.
PTO Customer No. 20985
4350 La Jolla Village Drive, Suite 500
San Diego, California 92122
Telephone: (858) 678-5070
Facsimile: (858) 678-5099

10156052.doc